Neural Networks 64 (2015) 12–18

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

2015 Special Issue

Measuring the usefulness of hidden units in Boltzmann machines with mutual information☆

Mathias Berglund *, Tapani Raiko, Kyunghyun Cho

Department of Information and Computer Science, Aalto University School of Science, Finland

ARTICLE INFO

Article history:
Available online 28 September 2014

Keywords:
Deep learning
Restricted Boltzmann machine
Deep Boltzmann machine
Pruning
Structural learning
Mutual information

ABSTRACT

Restricted Boltzmann machines (RBMs) and deep Boltzmann machines (DBMs) are important models in deep learning, but it is often difficult to measure their performance in general, or measure the importance of individual hidden units in specific. We propose to use mutual information to measure the usefulness of individual hidden units in Boltzmann machines. The measure is fast to compute, and serves as an upper bound for the information the neuron can pass on, enabling detection of a particular kind of poor training results. We confirm experimentally that the proposed measure indicates how much the performance of the model drops when some of the units of an RBM are pruned away. We demonstrate the usefulness of the measure for early detection of poor training in DBMs.

1. Introduction

Restricted Boltzmann machines (RBMs) and deep Boltzmann machines (DBMs) are important models in the field of *deep learning*, helping to achieve state-of-the-art performances in many machine learning tasks. However, both models are known to be difficult to train (Cho, Raiko, & Ilin, 2013; Cho, Raiko, Ilin, & Karhunen, 2013).

It is often possible to determine the success of training an RBM or a DBM by observing the performance of the features learned by the Boltzmann machine (BM) on another task. One important shortcoming of this approach is that we are only given the final and overall performance of a model. Furthermore, because either an RBM or a DBM is not trained directly to maximize the performance on another task, it is not clear how the final performance measured on that task is related to the actual training of the model.

In this paper we are more interested in the behavior of an RBM or a DBM during the training phase. We expect that it would be beneficial to have deeper understanding of the learning dynamics of a model beyond a mere final performance. Specifically, we are interested in how the statistical characteristics of each individual neuron in a BM evolves and how those can be collected during learning. We claim that collecting and investigating the statistics of individual neurons is one important way to understand the underlying factors affecting the learning dynamics of a model.

One potential application of observing the statistical characteristics of individual neurons of a BM is in learning its structure. The statistics of each neuron may reveal its usefulness as well as how saturated the whole model is, which may allow us to either prune useless neurons or add more neurons to the model. This approach of adaptively learning the structure of a model has been applied to other types of neural networks and probabilistic models (see, e.g., Adams, Wallach, & Ghahramani, 2010; Engelbrecht, 2001; Reed, 1993; Zhou, Sohn, & Lee, 2012) with the reported benefits of, for instance, a fewer number of hyperparameters (Adams et al., 2010), better generalization and faster performance (Reed, 1993).

In this paper, we propose that the mutual information between input data (clamped to the visible neurons) and each individual hidden neuron measures the statistical importance of each hidden neuron.

After reviewing RBMs and DBMs in Section 2, we describe the mutual information (MI) measure for evaluating the importance of individual hidden units of a BM in Section 3. Experimenting with RBMs in Section 4, we demonstrate the usefulness of the measure in pruning and adding neurons as well as visualizing the progress of learning. In Section 5, we compare a plain training algorithm with one using pretraining on DBMs using the proposed MI measure.

2. Boltzmann machines: background

2.1. Restricted Boltzmann machines

A restricted Boltzmann machine (RBM) (Smolensky, 1986) is a variant of a Boltzmann machine that has a bipartite structure

☆ This paper is an extended version of Berglund, Raiko, and Cho (2013).

* Corresponding author. Tel.: +358 503210978.
E-mail addresses: mathias.berglund@aalto.fi, m.p.e.berglund@gmail.com (M. Berglund).

http://dx.doi.org/10.1016/j.neunet.2014.09.004
0893-6080/

such that each visible neuron is connected to all hidden neurons and each hidden neuron to all visible neurons, but there are no edges between the same type of neurons (see Fig. 1 (left) for an illustration). An RBM defines the negative energy of each state $(\mathbf{x}, \mathbf{h})$ by

$$-E(\mathbf{x}, \mathbf{h} \mid \boldsymbol{\theta}) = \mathbf{b}^\top \mathbf{x} + \mathbf{c}^\top \mathbf{h} + \mathbf{x}^\top \mathbf{W} \mathbf{h},$$

and assigns the following probability to the state via Boltzmann distribution:

$$p(\mathbf{x}, \mathbf{h} \mid \boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \exp\left\{-E\left(\mathbf{x}, \mathbf{h} \mid \boldsymbol{\theta}\right)\right\}, \tag{1}$$

where $\boldsymbol{\theta} = \{\mathbf{b}, \mathbf{c}, \mathbf{W}\}$ is a set of parameters consisting of visible and hidden biases as well as weights between visible and hidden neurons. $Z(\boldsymbol{\theta})$ is a normalization constant that makes the probabilities sum up to one.

2.2. Deep Boltzmann machines

A Deep Boltzmann machine (DBM) was proposed by Salakhutdinov and Hinton (2009) as a structurally relaxed version of an RBM. A DBM stacks multiple additional layers of hidden neurons on the top of an RBM. As was the case with an RBM, consecutive layers are fully connected, while there is no edge among the neurons in each layer. See Fig. 1 (right) for an illustration of a typical DBM.

The negative energy function is defined as

$$-E(\mathbf{x}, \mathbf{h} \mid \boldsymbol{\theta}) = \mathbf{b}^\top \mathbf{x} + \mathbf{c}_{[1]}^\top \mathbf{h}_{[1]} + \mathbf{x}^\top \mathbf{W} \mathbf{h}_{[1]} + \sum_{l=2}^{L} \left(\mathbf{c}_{[l]}^\top \mathbf{h}_{[l]} + \mathbf{h}_{[l-1]}^\top \mathbf{U}_{[l-1]} \mathbf{h}_{[l]}\right),$$

where L is the number of hidden layers. The probability of each state is assigned as in Eq. (1). The states and biases of the hidden neurons at the lth hidden layer and the weight matrix between the lth and $(l+1)$th layers are respectively defined by

$$\mathbf{h}_{[l]} = \left[h_1^{[l]}, \ldots, h_{q_l}^{[l]}\right]^\top, \qquad \mathbf{c}_{[l]} = \left[c_1^{[l]}, \ldots, c_{q_l}^{[l]}\right]^\top,$$

$$\mathbf{U}_{[l]} = \left[u_{ij}^{[l]}\right],$$

where q_l is the number of the neurons in the lth layer and $\mathbf{U}_{[l]} \in \mathbb{R}^{q_l \times q_{l+1}}$.

2.3. Why Boltzmann machines?

The RBM is an important basic building block of deep neural networks. Hinton and Salakhutdinov (2006) showed that a multilayer perceptron (MLP) with many hidden layers can be trained well by greedily pretraining each pair of consecutive layers as an RBM. Furthermore, deep generative models such as deep belief networks and DBMs were found to be easily trainable if the parameters were initialized by greedy layer-wise pretraining using an RBM (Hinton, Osindero, & Teh, 2006; Salakhutdinov & Hinton, 2009). A DBM was also found to be effective at initializing the parameters of an MLP as well (Salakhutdinov & Hinton, 2009, 2012).

Both RBMs and DBMs have been found to be useful on their own as well. Salakhutdinov, Mnih, and Hinton (2007) showed that an RBM can be used for collaborative filtering. Recently Srivastava and Salakhutdinov (2012) showed that a DBM is good at extracting shared representation of multiple modalities of data. Furthermore, Lee, Grosse, Ranganath, and Ng (2009) showed that a stack of convolutional RBMs is able to extract the hierarchical part-based features in unsupervised way. However, all these achievements by RBMs and DBMs require that these neural networks were trained *well*.

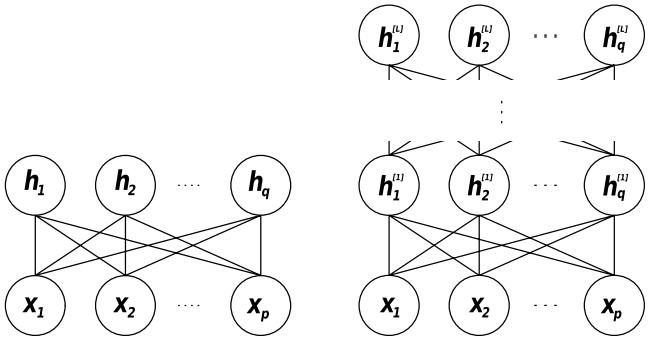

Fig. 1. Left: Example structure of an RBM. Right: Example structure of a DBM.

It is unfortunately not easy to monitor the progress of training BMs. It is not even easy to evaluate whether the model as a whole has trained well. The most direct measure for evaluating a BM is the log-likelihood of a model which is maximized when training a BM. However, due to the computational intractability of the normalization constant, it is not feasible to use the log-likelihood during training. Although it is difficult to obtain even approximations to the normalization constant (Salakhutdinov, 2008), there are methods for estimating it with reasonable complexity (Desjardins, Bengio, & Courville, 2011). However, the log-likelihood does not provide any detailed description on the contribution of each subpart of a BM, which is important if we were to adapt the model on-the-fly. One strategy for obtaining that information would be to estimate the log-likelihood for models where one hidden neuron is removed in each estimate, and the biases updated to compensate for the average activation of the missing neuron. Although potentially a useful measure, it would be rather costly to compute.

Some monitors have been proposed for visualizing the progress of training from some perspective. One method notable for monitoring individual neurons is a visual approach where neuron activation probabilities are plotted. In the method, one plots 2D-images of activation probabilities of mini-batches where the axes represent neurons and samples, respectively (Hinton, 2010). These images are useful for monitoring how individual neurons perform, and whether there are difficult individual samples in the data set. However, these images become impractical if the number of hidden neurons becomes too large.

Hence, in this paper we explore a relatively cheap measure that can be used to evaluate the contribution of each hidden neuron in an RBM and a DBM without becoming impractical even for large models.

3. Mutual information measure

Neural networks such as multilayer perceptrons (MLP) are often criticized for being a *black box*. That is, it is difficult to understand what the subparts of a model, specifically the individual neurons, are doing. The variance of each hidden neuron has been one choice of measure for evaluating the effect of each hidden neuron (see, e.g., Glorot & Bengio, 2010). The underlying rationale is that any neuron with constant activation across different inputs cannot convey any discriminative information about input samples.

The variance of each hidden neuron is, however, not applicable to neural networks having stochastic hidden neurons such as Boltzmann machines (BM). The variance of the activations of a single hidden neuron can be large, even though the probability of the neuron is constant across different inputs. In other words, the variance of a stochastic neuron is not an appropriate measure of the (discriminative) information conveyed by the neuron.

We therefore propose to measure the *relevant* activity (or importance) of a single hidden neuron h_j in BMs by measuring the

mutual information (MI) between the observation vector (or the set of the activations of visible neurons) $\mathbf{x}$ and the hidden neuron h_j. Specifically, the MI-measure of the hidden unit MI_j is

$$\begin{aligned}\mathrm{MI}_j &= \sum_{h_j\in\{0,1\}}\sum_{\mathbf{x}} P(\mathbf{x}, h_j)\log_2\left(\frac{P(h_j \mid \mathbf{x})}{P(h_j)}\right)\\ &= \sum_{h_j\in\{0,1\}}\Bigg[-P(h_j)\log_2\left(P(h_j)\right)\\ &\quad + \sum_{\mathbf{x}} P(\mathbf{x})P(h_j \mid \mathbf{x})\log_2\left(P(h_j \mid \mathbf{x})\right)\Bigg]\\ &\approx \sum_{h_j\in\{0,1\}}\sum_{t=1}^{T}\frac{1}{T}P(h_j \mid \mathbf{x}_t)\Bigg[-\log_2\left(\sum_{t=1}^{T}\frac{1}{T}P(h_j \mid \mathbf{x}_t)\right)\\ &\quad + \log_2\left(P(h_j \mid \mathbf{x}_t)\right)\Bigg] \qquad (2)\end{aligned}$$

where we have used an empirical estimate for $P(h_j)$ in the final step.

We use the logarithm with base 2 in order to get the amount of information as bits. It is easy to show that the mutual information between the binary hidden neuron h_j and the visible neurons $\mathbf{x}$ ranges from 0 to 1 bit and defines the upper bound of the average information the hidden neuron can convey about the state of the visible neurons $\mathbf{x}$.

The major part of the computational complexity of the measure is the computation of $P(h_j \mid \mathbf{x}_t)$. However, they are computed during training and validation anyway, so the additional computational cost of the MI-measure is marginal.

Note that the MI-measure cannot measure how well the hidden unit works together with the other hidden units. This makes it impossible to use the proposed MI-measure as a training criterion, since it may result in all hidden neurons learning identical features. However, the MI-measure serves as an upper bound on how useful the hidden neuron can be for a task the BM is trained for. In other words, it is possible to tell whether a hidden neuron is useless for another task by observing its MI-measure. If the MI-measure of a hidden neuron is close to zero, the hidden neuron will not be useful for any further task, as it does not contain any information on inputs. This becomes a particularly useful measure when training BMs, as we empirically show later that certain common training situations yield hidden neurons with very low MI-measures.

4. Experiments on restricted Boltzmann machines

This section studies the use of MI-measure as a measure of the usefulness of hidden neurons in the case of RBMs by (1) pruning, (2) adding new hidden neurons during training, and (3) visualizing the progress of training.

4.1. Pruning neurons after training

We evaluate the contribution of hidden neurons with varying MI-measures to a simple classification task. We train an RBM with 4000 hidden neurons on the handwritten digits data set (MNIST, LeCun, Bottou, Bengio, & Haffner, 1998) and use the hidden neuron activations as inputs to a logistic regression classifier. We then prune 100 hidden neurons at a time in the order of the MI-measures. We do this both in the ascending and descending order, in addition to randomly pruning 100 hidden neurons at a time. The activations of the remaining hidden neurons are then used as features for the classifier. The RBM is trained with the adaptive learning rate (Cho, Raiko, & Ilin, 2011).

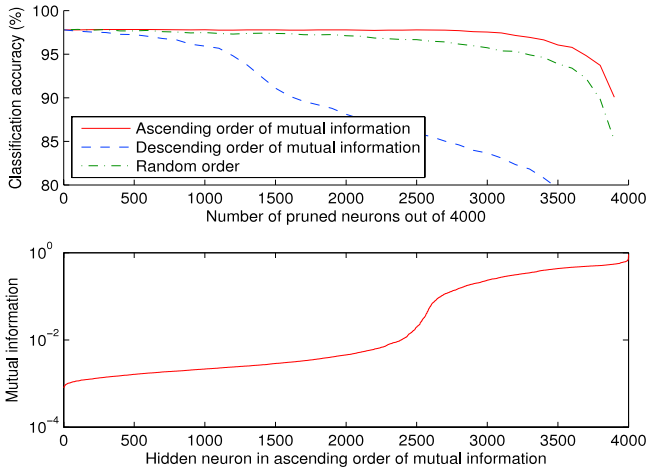

Fig. 2. The evolution of the classification accuracy while pruning hidden neurons. The original RBM with 4000 hidden neurons is trained for 1000 epochs.

The results are shown in Fig. 2. As predicted, the classification performance does not drop markedly in the beginning of pruning neurons in the ascending order of MI-measure, when the pruned neurons have a very low MI-measure. On the other hand, when pruning is done in the descending order of MI-measure, the classification performance drops significantly faster. Pruning in random order which is bounded from above and below by the ascending and descending orders of MI-measure, also resulted in significantly faster drop compared to pruning in the ascending order.

When illustrating the filters learned by the 100 neurons with the highest and lowest MI-measure, respectively, we see in Fig. 4 that the neurons with a high MI-measure also visually look like filters learned by a well-trained RBM, while the neurons with a low MI-measure are considerably more noisy.

We also run a similar experiment with an RBM having 1000 hidden neurons which are trained using the enhanced gradient (Cho, Raiko, & Ilin, 2013). The critical difference to the standard gradient is that weight matrix element updates are proportional to the covariance between visible and hidden neurons instead of the dot-product of their activations. The enhanced gradient is less prone to hidden layers learning similar features, and has empirically been shown to yield more hidden neurons learning useful features. The results of the experiment clearly differ from the previous experiment, in that the drop in classification performance is much less dependent on whether the pruning is done in the ascending, random or descending orders of MI-measure.

As can be seen from Fig. 3, this model differs from the previous in that all the hidden neurons have a fairly high MI-measure. Therefore, no hidden neuron has a low enough MI-measure as to not be able to convey enough information about the observation $\mathbf{x}$. This clearly reveals that high MI-measure is not always an accurate indicator of high significance for the classification task—only a sufficiently low MI-measure can with confidence predict that a neuron is not useful. In other words, if the MI-measures of most of the hidden neurons are high enough, one should be cautious in using the MI-measure as an indicator of the importance of each of those hidden neurons.

4.2. Adding neurons during training

One potential way to learn an optimal structure of an RBM would be to add neurons to the hidden layer in the middle of training. This has been studied by, for instance, Adams et al. (2010) and Zhou et al. (2012) in the context of denoising autoencoders

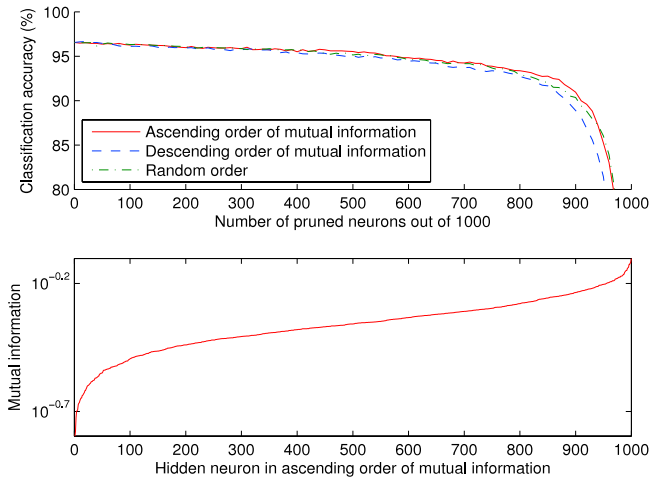

Fig. 3. The evolution of the classification accuracy while pruning hidden neurons. The original RBM with 1000 hidden neurons is trained for 100 epochs using the enhanced gradient (Cho, Raiko, & Ilin, 2013).

and deep belief networks. Here, we investigate the effect of adding more hidden neurons in the middle of training an RBM.

It is, however, possible that adding neurons to a layer of hidden units where the previous hidden units have been trained for some time would not be beneficial, as the added neurons might not learn relevant structures in addition to the already co-adapted hidden neurons. In order to test that hypothesis, we train an RBM of 500 hidden neurons for 50 epochs, after which we add another set of 500 hidden neurons to the model and train it for another 50 epochs. When we add neurons, we reset the adaptive learning rate to the initial values.

Fig. 5 shows the development of the MI-measures. Clearly the MI-measures of the newly added hidden neurons stay considerably lower than those of the initial 500 hidden neurons. It is also worth noting that the MI-measures of the added hidden neurons do not increase over training, unlike those of the initial hidden neurons.

We further use the hidden neuron activations as features for a logistic regression classifier and compare the performance to two RBMs that are trained from start to end with 500 and 1000 hidden units, respectively. For comparability, we also reset the

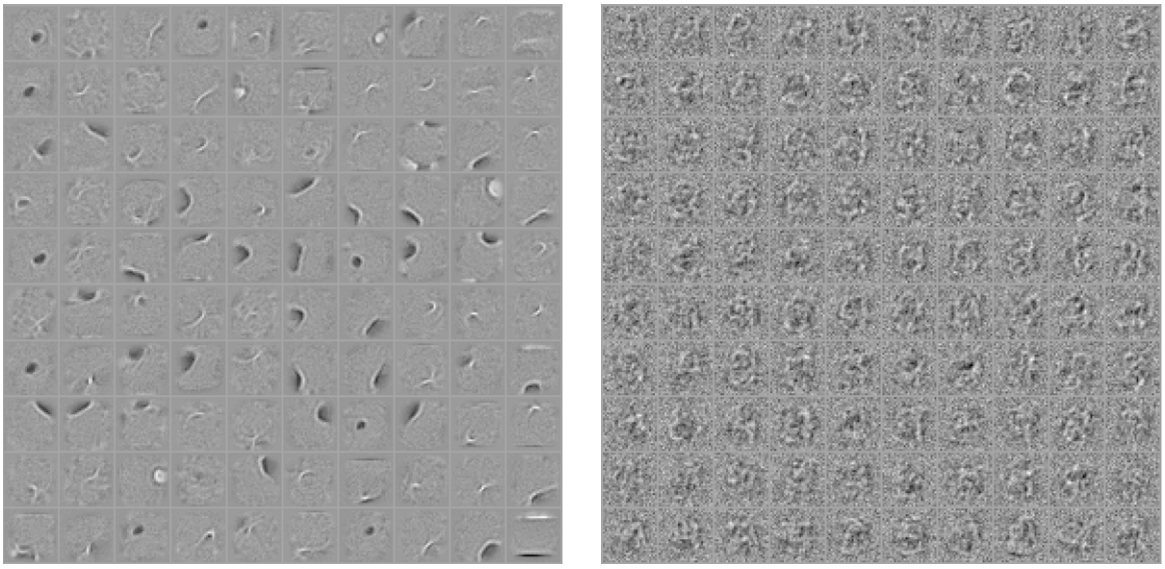

Fig. 4. Illustration of the filters learned by the 100 neurons with the highest MI-measure (left) and the lowest MI-measure (right) for an RBM with 4000 hidden units.

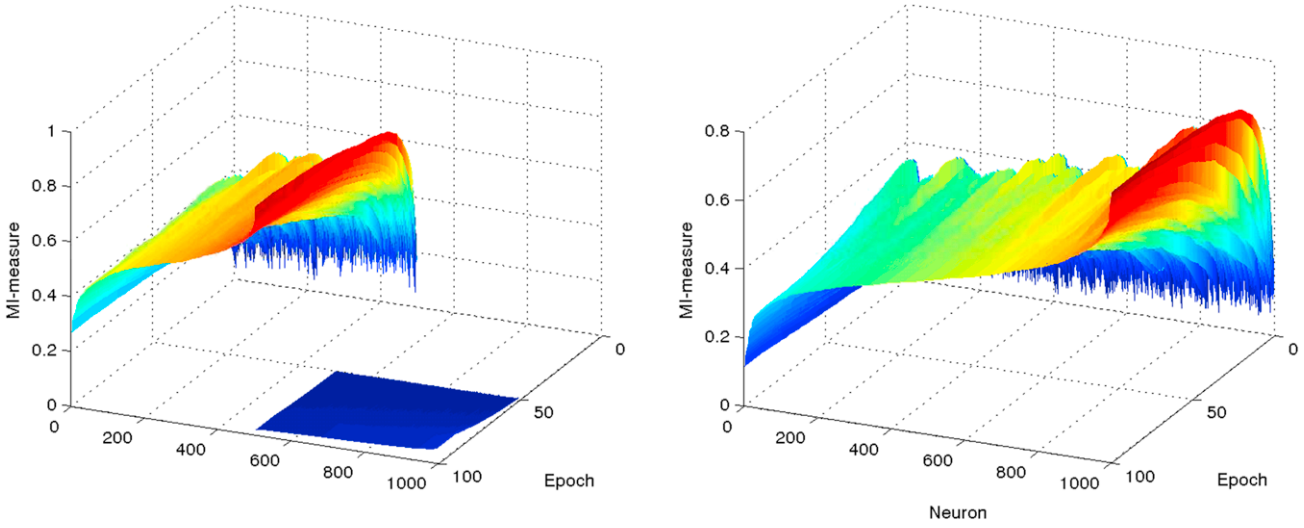

Fig. 5. The MI-measures of the hidden neurons of the RBM where 500 hidden neurons are added in the middle of the training (left) vs. those of the hidden neurons of the RBM with 1000 hidden neurons from the beginning of training (right). Both models are trained for 100 epochs.

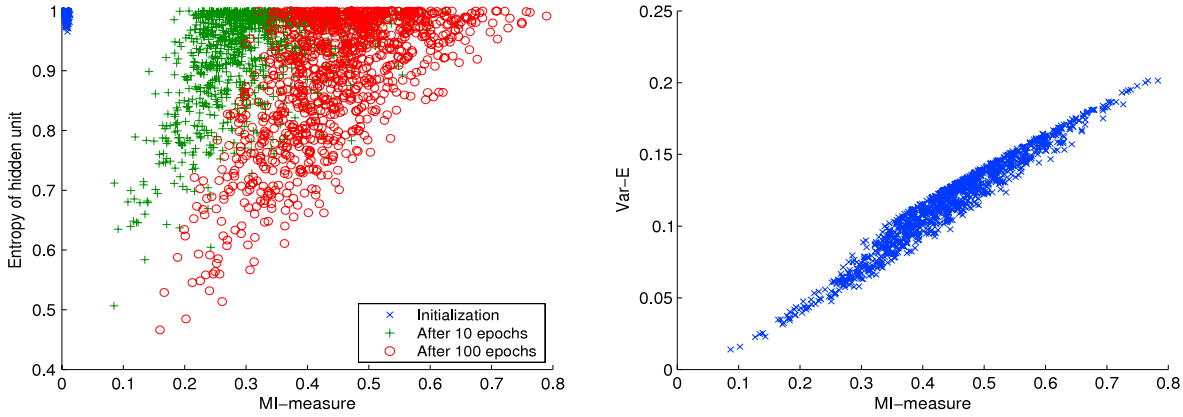

Fig. 6. (Left) The MI-measure vs. the entropy of each hidden neuron at different stages of learning. (Right) The variance of the expected activation (Var-E) vs. the MI-measure of each hidden neuron after 100 epochs.

Table 1
Classification accuracies obtained by the features learned by the RBM with added hidden neurons and the normal RBM. Reported ranges are standard errors.

Begin units	500	500	1000	1000
Added units	–	500	–	–
Epochs	100	100	100	50
Accuracy	95.17 ± 0.08%	95.35 ± 0.19%	96.38 ± 0.08%	96.43 ± 0.14%

adaptive learning rate to the initial values after 50 epochs for the two benchmark models.

Table 1 lists the mean accuracy from ten runs. The performance of the model with added features is clearly inferior to the model trained with 1000 hidden neurons for 100 epochs. Even if the 1000 neuron model is only allowed to train for 50 epochs, it still performs better than the model where we add 500 neurons in the middle of the training and continue training for 50 epochs.

This result suggests that the proposed MI-measure reflects well the amount of information carried by the hidden neurons. From this, we may further hypothesize that it is not easy for the newly added hidden neuron to learn an informative feature and is important to have a way to initialize the newly added neurons (Zhou et al., 2012).

4.3. Relations to entropy and variance

By investigating the definition of the proposed MI-Measure in Eq. (2), we find two related measures. The first one is the entropy which is defined by

$$H(h_j) = -\sum_{h_j \in \{0,1\}} P(h_j) \log_2 \left(P(h_j)\right).$$

This is exactly the first term of the MI-measure, which corresponds to measuring the amount of information learned by each hidden neuron without considering the relationship with respect to observations.

The second measure is the variance of the expected activation, which we call Var-E. The Var-E is defined as

$$\text{Var-E}_j = \text{Var}_t \left(E_{h_j|\mathbf{x}_t}\left[h_j\right]\right).$$

This measure was used by Cho, Raiko, and Ilin (2013) to check if the activation probability of a hidden neuron of an RBM is dependent on the input. This may be considered in analogy to the second term of the MI-measure in Eq. (2), considering that the Var-E also measures the relationship between the hidden neuron and the input.

In Fig. 6, we show the relationship among these measures in an RBM with 1000 hidden neurons. As expected from the fact that the MI-measure reflects the information *of the input* conveyed by a hidden neuron, it is clear that the Var-E is more correlated with the MI-measure than the entropy is. If used as a diagnostic measure to detect neurons with near-zero MI-measure, Var-E can be used as a proxy for the MI-measure in cases where such implementation is easier.

5. Experiments on deep Boltzmann machines

Although deep Boltzmann machines (DBM, Salakhutdinov & Hinton, 2009) have been used with great success in several applications (see, e.g. Srivastava & Salakhutdinov, 2012), they are generally considered difficult to train (Cho, Raiko, Ilin, & Karhunen, 2013; Goodfellow, Mirza, Courville, & Bengio, 2013). One method for alleviating that difficulty is pretraining.

In this section, we briefly show that the proposed MI-measure can be used efficiently to detect the failure of training DBMs. To illustrate this point, we train three DBMs with two layers of 1000 hidden neurons for 100 epochs: the first using "vanilla" Persistent Contrastive Divergence (Tieleman, 2008), the second one with two-stage pretraining (Cho, Raiko, Ilin, & Karhunen, 2013) and the third one with the adaptive learning rate[1] (Cho et al., 2011). We then use a mean-field approximation for the hidden neuron activations. The hidden neuron activations of both layers of all of the models are used as features for a logistic classifier.

We achieved an accuracy of 96.6% with pretraining, 93.7% with the adaptive learning rate and 11.3% without pretraining or adaptive learning rate. The near-chance performance of the last case represents a failure case of training DBMs. In Fig. 7 we have illustrated the MI-measure development during training for the cases with pretraining and without either pretraining or adaptive learning rate. We can clearly see that in the latter case of training failure, the mutual information of, especially, the second layer is extremely low. The mutual information is in fact so low, that the entire second layer only conveys on average a maximum of 1.5×10^{-10} bits of information about the observations in the training set. This illustration suggests that by observing the proposed MI-measure we can detect if the DBM does not learn useful features of the data.

If we examine the MI-measure of both hidden layers at the end of training for all the three training cases, we can see in Fig. 8 that without pretraining but with the adaptive learning rate the

[1] We limit the learning rate never to exceed 0.0001.

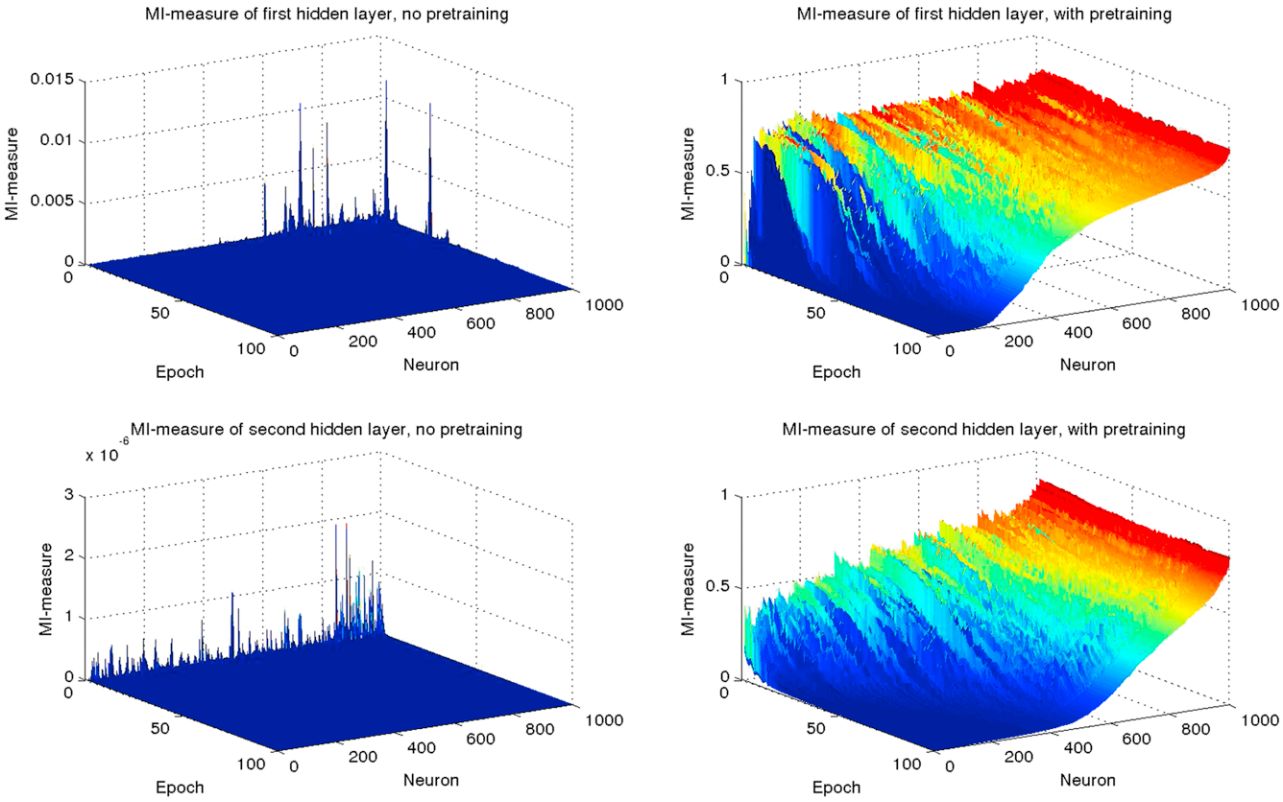

Fig. 7. The MI-measures of the hidden neurons of the DBMs having two layers of hidden units trained without pretraining (left) and with pretraining (right).

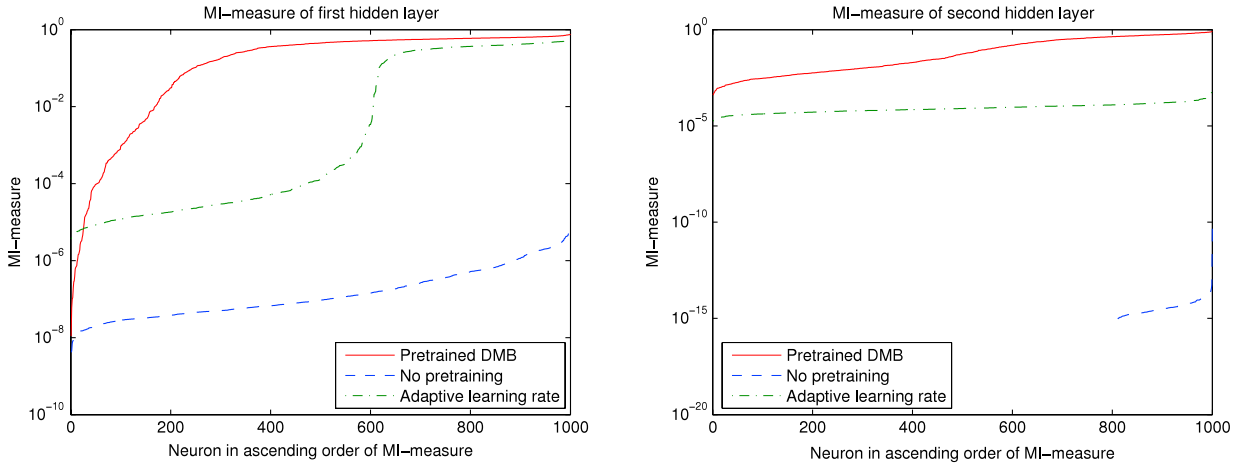

Fig. 8. The MI-measures of the hidden neurons of the DBM first hidden layer (left) and second hidden layer (right) at the end of training for different training options.

DBM training does not fail completely. However, we can see that almost half of the neurons in the first hidden layer and all of the neurons in the second hidden layer have a very low MI-measure, which is in line with the classification performance. Although we do not propose to use the MI-measure as a sole criterion for model selection, it can serve as an indicator for how well subparts of the model have been trained.

6. Discussion

We propose to use the mutual information between the observation vector and a single hidden unit (MI-measure) for evaluating the importance of individual hidden neurons of a Boltzmann machine (BM). We expect that observing the proposed MI-measure can be useful during training to determine hidden neurons that are not learning any useful features.

The proposed measure is a useful monitoring tool for training BMs. The measure is therefore likely to be suitable to speed up development of training algorithms for BMs. In addition, it could be used to find better methods for pruning or adding neurons.

By observing the proposed MI-measures of the hidden neurons of BMs, we were able to notice a number of cases where many hidden neurons become non-informative. Firstly, training a large RBM with the traditional gradient may include a lot of inactive neurons. Secondly, when an RBM has already learned the important structure of data, we found that it is difficult to add new neurons. Thirdly, we found that the hidden neurons in the deeper layers of a deep Boltzmann machine (DBM) may easily become useless with standard learning methods used to train RBMs.

The proposed MI-measure, however, has some limitations. Firstly, the MI-measure, especially in the case of RBMs, effectively ignores the interaction, or co-adaptation, among multiple hidden

neurons. This is problematic, since BMs are known to produce very distributed representation of data. The measure is therefore not suitable as the sole criterion for e.g. model selection. Secondly, the MI-measure which was shown to correlate well with the entropy does not agree with the well known and observed phenomenon that sparse features which have in general low entropy are good for many machine learning tasks including classification. This suggests that a more accurate measure may be designed by combining the proposed MI-measure and the entropy, in the future.

Acknowledgment

This work was supported by the Academy of Finland (Finnish Centre of Excellence in Computational Inference Research COIN, 251170).

References

Adams, R. P., Wallach, H. M., & Ghahramani, Z. (2010). Learning the structure of deep sparse graphical models. In *JMLR workshop and conference proceedings*: *Vol. 9. Proceedings of the thirteenth international conference on artificial intelligence and statistics*, (AISTATS 2010), JMLR W&CP.

Berglund, M., Raiko, T., & Cho, K. (2013). Measuring the usefulness of hidden units in Boltzmann machines with mutual information. In *Proceedings of the 20st international conference on neural information processings*, (ICONIP). November.

Cho, K., Raiko, T., & Ilin, A. (2011). Enhanced gradient and adaptive learning rate for training restricted Boltzmann machines. In *Proceedings of the 28th international conference on machine learning*, (ICML 2011). (pp. 105–112). New York, NY, USA: ACM.

Cho, K., Raiko, T., & Ilin, A. (2013). Enhanced gradient for training restricted Boltzmann machines. *Neural Computation*, *25*(3), 805–831.

Cho, K., Raiko, T., Ilin, A., & Karhunen, J. (2013). A two-stage pretraining algorithm for deep Boltzmann machines. In *Proceedings of the 23rd international conference on artificial neural networks*, (ICANN 2013). September.

Desjardins, G., Bengio, Y., & Courville, A. C. (2011). On tracking the partition function. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, & K. Weinberger (Eds.), *Advances in neural information processing systems 24* (pp. 2501–2509). Curran Associates, Inc.

Engelbrecht, A. P. (2001). A new pruning heuristic based on variance analysis of sensitivity information. *Transactions on Neural Networks*, *12*(6), 1386–1399.

Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *JMLR workshop and conference proceedings*: *Vol. 9. Proceedings of the thirteenth international conference on artificial intelligence and statistics*, (AISTATS 2010), (pp. 249–256). JMLR W&CP.

Goodfellow, I., Mirza, M., Courville, A., & Bengio, Y. (2013). Multi-prediction deep Boltzmann machines. In *Advances in neural information processing systems* (pp. 548–556).

Hinton, G. (2010). A practical guide to training restricted Boltzmann machines. *Momentum*, *9*(1), 926.

Hinton, G., Osindero, S., & Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, *18*(7), 1527–1554.

Hinton, G., & Salakhutdinov, R. (2006). Reducing the dimensionality of data with neural networks. *Science*, *313*(5786), 504–507.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*, 2278–2324.

Lee, H., Grosse, R., Ranganath, R., & Ng, A. (2009). Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th annual international conference on machine learning*, (ICML 2009). (pp. 609–616). New York, NY, USA: ACM.

Reed, R. (1993). Pruning algorithms—a survey. *Transactions on Neural Networks*, *4*(5), 740–747.

Salakhutdinov, R. (2008). *Learning and evaluating Boltzmann machines. Tech. Rep. UTML TR 2008-002*. Department of Computer Science, University of Toronto.

Salakhutdinov, R., & Hinton, G. (2009). Deep Boltzmann machines. In *JMLR workshop and conference proceedings*: *Vol. 5. Proceedings of the twelfth internation conference on artificial intelligence and statistics*, (AISTATS 2009), (pp. 448–455). JMLR W&CP.

Salakhutdinov, R., & Hinton, G. (2012). An effcient learning procedure for deep Boltzmann machines. *Neural Computation*, *24*, 1967–2006.

Salakhutdinov, R., Mnih, A., & Hinton, G. (2007). Restricted Boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on Machine learning*, (ICML 2007). (pp. 791–798). New York, NY, USA: ACM.

Smolensky, P. (1986). Information processing in dynamical systems: foundations of harmony theory. In *Parallel distributed processing: explorations in the microstructure of cognition, vol. 1: foundations* (pp. 194–281). Cambridge, MA, USA: MIT Press.

Srivastava, N., & Salakhutdinov, R. (2012). Multimodal learning with deep Boltzmann machines. In P. Bartlett, F. Pereira, C. Burges, L. Bottou, & K. Weinberger (Eds.), *Advances in neural information processing systems 25* (pp. 2231–2239).

Tieleman, T. (2008). Training restricted Boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the 25th internation conference on machine learning*, (ICML 2008). (pp. 1064–1071). New York, NY, USA: ACM.

Zhou, G., Sohn, K., & Lee, H. (2012). Online incremental feature learning with denoising autoencoders. In *JMLR workshop and conference proceedings*: *Vol. 22. Proceedings of the fifteenth internation conference on artificial intelligence and statistics*, (AISTATS 2012), (pp. 1453–1461). JMLR W&CP.